CSCE 561 Assignment #1, Fall 2019

Dr. Vijay V Raghavan

Assigned: September 18, 2019

Due: October 02, 2019

(TOPICS: Creation and Processing of Inverted Index, Boolean and Fuzzy set based Retrieval models)

Note:

1. *You must show all details of work for each question*
2. *Staple the question and answer sheet together*
3. *Make a cover with Name, CLID*
4. *Number all pages and give an index to each question*
5. *Most importantly, any sort of cheating will **NOT** be tolerated. More information can be found on class Web page on cheating policy.*

Q1. (10 Points) INVERTED INDEX

Consider these documents:

Doc 1: breakthrough in drug discovery with AI

Doc 2: cancer candidate drug identified using AI

Doc 3: We are turning the cancer drug-discovery paradigm upside down with AI models

Doc 4: new hopes for cancer patients

a. Draw the inverted index representation for this collection.

b. For the document collection shown above, what are the returned results for these queries:

(i). cancer AND drug

(ii). cancer AND NOT(discovery OR AI)

NOTE**:** Punctuation and Stopwords are not stored in the dictionary of terms. Use the link mentioned in Q4 for the list of Stopwords.

Q2. (20 points) QUERY OPTIMIZATION

A. What is the best order for query processing of below queries:

(i) t_1 *AND* t_2 *AND* t_3

(ii) (t_4 *OR* t_5) *AND* (t_6 *OR* t_7) *AND* (t_8 *OR* t_9)

Note: State any assumptions you are making to support your answers.

B. Recommend a query processing order for

(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes),

given the following postings list sizes:

Term	Postings size
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Table 1: Terms and posting list sizes

C. We can use distributive laws for AND and OR to rewrite queries.

(i) Show how to rewrite the query below into disjunctive normal form using the distributive laws.

(Brutus OR Caesar) AND NOT (Antony OR Cleopatra)

(ii) Would the resulting query be more or less efficiently evaluated than the original form of this query?

(iii) Is this result true in general or does it depend on the words and the contents of the document collection?

Q3. (30 points) Table 2 represents a matrix of documents and terms. Each document is represented by a row. Each element in the row represents the weights of importance (i.e. membership values) of corresponding terms in that document. Compute the RSVs of the following queries for each of the documents d_1 through d_3

	t_1	t_2	t_3	t_4	t_5
d_1	0.6	0.2	0.6	0.9	0.7
d_2	0.4	0.4	0.0	0.4	0.5
d_3	0.8	0.5	0.9	0.1	0.6

Table 2: term-document Matrix

a) Assume Λ and V are defined, respectively, as min and max.

(1) $(t_2 \Lambda \neg t_5) V (t_1 \Lambda t_3)$

(2) $(t_2 \Lambda t_3 \Lambda t_4) V (\neg t_1 \Lambda \neg t_4) V (t_1 \Lambda \neg t_5)$

b) Suppose V and Λ defined as 'bold' combination operators:

$a V b = \min (a + b, 1)$,

$a \Lambda b = \max (0, a + b - 1)$

(3). $\neg t_1 \Lambda t_4$

(4). $(t_2 \Lambda \neg t_5) V (\neg t_3 \Lambda t_4) V t_1$

c) Repeat part a) using 0.3-level fuzzy sets. Compare the rankings of documents produced by these two methods and explain how the use of level fuzzy sets affects the retrieval results.

Q4. (50 Points) PROGRAMMING

This is a programming assignment where you design and implement your own **information retrieval system.**

This project contains two parts. In Part 1, you will build an indexing component, where a large collection of documents is indexed into a searchable, persistent data structure. In Part 2, you will add a searching component, using vector space model **(Part 2 will be turned in as a part of assignment #2)**.

Part 1 (20 Points)

Read a set of files, parse them into documents and terms, and produce data structures associated with the inverted index. These data structures will be used in part 2. The program needs to save several data structures to disk, at minimum; these include the dictionary (table of words occurring in the text, appropriate metadata/statistics and pointers to inverted index), the document list (with links to original text) and inverted index itself.

Note: During query processing, the Dictionary part of the Inverted File structure will reside in the main memory.

In order to fully implement the vector space retrieval model you may need to retain several pieces of information about the documents in the dictionary and the postings lists of the inverted index data structures such as the total number of documents, the term weight values for each document, etc. Try to store values that you anticipate are needed to calculate the retrieval status values that can only be generated with high run-time computational cost, if not precomputed and stored. In other words, writing out the necessary intermediate results from Part 1 and reading back into Part 2 may be a better approach than re-computing everything from scratch in memory each time.

Resources

Here are the resources needed for Part 1.

Corpus: Corpus for this assignment is the Cranfield document collection, a set of 1398 abstracts from aerodynamics journal articles. This historically significant dataset was first used in the field of IR for accurate measurement of retrieval performance. A description of the corpus follows:

- cran.all: The Cranfield collection corpus file with many fields. The title (T) and the content of the abstract (W) are the two fields that you may use in this assignment.
- cran.qry: A set of 225 queries that can be used to test the retrieval performance of your system.
- cranqrel: Query relevance judgements for each query. Each query is accompanied with a set of documents that are relevant in separate lines.

  Note: qrels.text is not needed now. It is used in future for system evaluation purposes.

- readme.txt: Fine grained description of each column in the dataset files.

Download Link: http://ir.dcs.gla.ac.uk/resources/test_collections/cran/

Stopwords: (You can use this link for stopword removal in Q1)
http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

Stemmer: http://tartarus.org/~martin/PorterStemmer/

Steps

1) Parse the files containing the documents into individual documents (if necessary) and words

2) Remove stop words and stem (using the Porter stemmer) the remaining words

3) Invert the collection in memory (Each postings entry should have a Doc-ID and the associated term frequency value)

4) Store the data structures associated with the inverted file

Deliverables

1) Corpus Statistics:

- How many documents are in the collection?
- How many words are in the collection and how many unique words are in the
- indexing vocabulary of the collection?
- How many postings (inverted entries) are created for this collection, the length of the shortest and longest postings lists? Average length of the postings lists?

2) Inverted Index Statistics:

- How much time did it take to index the collection?
- How much disk space was consumed to store the inverted index of the collection and related data structures (including the dictionary, document list)?
- What is the size of the inverted index relative to the size of dataset?

Part 2 (to be turned in along with Assignment #2) (30 Points)

A program that accepts any query from the query file and searches for documents based on the data structures from Part 1.

Implement the search algorithm based on vector space model of retrieval. Your program should be able to return ranked results for a query. The program should display up to 20 results in descending order of their RSV values along with the total number of results (non-zero RSV values), snippet of text from document and time taken to retrieve those results.

Weighting Function

Use normalized frequency based on weight of term i in the document j is calculated as

$\mathbf{tf_{ij} = f_{ij} / max_k f_{kj}}$, Where $1 \leq k \leq M$, M is the total number of words in document j and f_{ij} is the frequency of i^{th} term in j^{th} document.

Steps

1) Implement Vector Space retrieval model and inverted index based search algorithm

2) User should be able to select a query interactively and obtain ranked results

Deliverables

Code and accompanying documentation: Your code needs to be well documented with comments. The comments should not literally verbalize what the code is doing; instead, it should provide a high-level summary of what the code is supposed to accomplish, listing aspects such as tacit assumptions and invariants (if you are using sophisticated data structures). You also need to include a README.txt file that describes how to compile your program and run it on various datasets.

Implementation

You may code your project in any programming language (such as C#, C++, Java, Python, Perl,VB, R).